ARMY RESEARCH LABORATORY

A Survey of Photon Mapping for Realistic Image Synthesis

by Justin L. Shumaker

ARL-TR-3608 **September 2005**

NOTICES

Disclaimers

Army Research Laboratory
Aberdeen Proving Ground, MD 21005-5068

ARL-TR-3608 **September 2005**

A Survey of Photon Mapping for Realistic Image Synthesis

Justin L. Shumaker
Survivability/Lethality Analysis Directorate, ARL

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	September 2005
2. REPORT TYPE	Final
3. DATES COVERED (From - To)	1 October 2004–1 February 2005
4. TITLE AND SUBTITLE	A Survey of Photon Mapping for Realistic Image Synthesis
5a. CONTRACT NUMBER	
5b. GRANT NUMBER	
5c. PROGRAM ELEMENT NUMBER	
6. AUTHOR(S)	Justin L. Shumaker
5d. PROJECT NUMBER	1L162618AH80
5e. TASK NUMBER	
5f. WORK UNIT NUMBER	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	U.S. Army Research Laboratory ATTN: AMSRD-ARL-SL-BE Aberdeen Proving Ground, MD 21005-5068
8. PERFORMING ORGANIZATION REPORT NUMBER	ARL-TR-3608
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	
10. SPONSOR/MONITOR'S ACRONYM(S)	
11. SPONSOR/MONITOR'S REPORT NUMBER(S)	
12. DISTRIBUTION/AVAILABILITY STATEMENT	Approved for public release; distribution is unlimited.
13. SUPPLEMENTARY NOTES	

14. ABSTRACT

This report is an overview and discussion of the finer details not found in most of the photon mapping documents that currently exist. While the vast majority of photon mapping media tends to market this global illumination algorithm as a do-it-all turn-key solution, they fail to discuss the finer details and pitfalls that developers will encounter in the implementation process. Finer details are also covered to not only explain what to expect during the implementation process, but also to provide graphical examples that may serve as a beacon during the various milestones throughout the implementation process.

15. SUBJECT TERMS

photon, mapping, image, synthesis, graphics, 3-D

16. SECURITY CLASSIFICATION OF: a. REPORT	b. ABSTRACT	c. THIS PAGE	17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL	26	Justin L. Shumaker

19b. TELEPHONE NUMBER (*Include area code*)
(410) 278-2834

Standard Form 298 (Rev. 8/98)
Prescribed by ANSI Std. Z39.18

Contents

List of Figures

1. Introduction

Global illumination is an inherent part of the realistic image synthesis capabilities of any modern rendering system. Traditionally, much computational work has been required to accurately simulate the interaction of light and a given environment. In 1995, a paper published by Jensen and Christensen (*1*) discussed how one could simulate global illumination with only a fraction of the computational cost when compared with Kajiya's path-tracing algorithm (*2*), which samples irradiance by tracing a large number of paths. Unlike radiosity, the photon mapping algorithm has the innate ability to accurately simulate catacaustics and diacaustics found in most reflective and refractive materials.

1.1 Purpose

As is the case with many graphics algorithms that attempt to speed up a given process, it is rather common to find degeneracies or even bias in the results those algorithms produce. Thus, when selecting a global illumination algorithm, one should consider the application in which it will be used, the amount of deviation from the correct result that is deemed acceptable, and the overhead associated with the complexity of using the algorithm in the application, i.e., the interface. Additionally, one should have an unbiased overview of what to expect in terms of implementation efforts and results to allow the reader to formulate their own opinions with respect to photon mapping. During the implementation of photon mapping into the BRL-CAD* raytracer "rt," the only photon mapping sources available were a few papers, a book by H. W. Jensen (*3*), and a handful of incomplete implementations done by various university students. None of these sources provided an overview that covered the implementation efforts and problems one might expect to encounter. The purpose of this report is to provide those considering photon mapping as a global illumination solution in their application with an overview of the implementation process, some of the hurdles they can expect to encounter, and methods to compensate for the various degeneracies that crop up. Each of these problematic areas will be enumerated by a bold number enclosed in square brackets, e.g., **[X]**.

1.2 Applications of Global Illumination and Photon Mapping

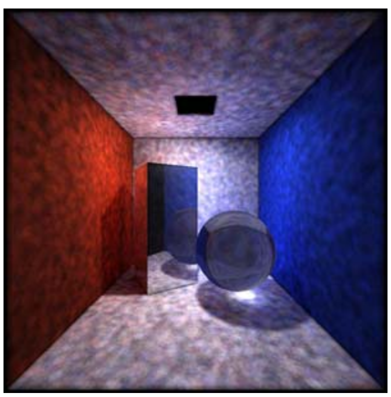
Global illumination is used in a variety of applications including energy transport, interactive graphics simulations, and image synthesis. Photon mapping caters well to applications that are requiring a form of global illumination at minimal additional computational cost. Photon mapping caters to applications that require both static and dynamic global illumination. Typical photon mapping code will vary on average from 500 to 2000 lines of code. The number of lines will be dependant on the number of heuristics the developer implements to reduce the complexity of the interface for the end user. While photon mapping requires little CPU power

* BRL-CAD is a registered trademark of the U.S. Army Research Laboratory.

relative to other global illumination algorithms such as path tracing, there exists a need to tune various functions within the code to deal with the heuristics and degeneracies that will need to be addressed.

2. Overview

From a top-level perspective, the foundation of photon mapping consists of two passes—building the photon map and using it to render. More specifically, a photon map is built during the first pass, which, in turn, is used to generate an irradiance cache (explained in more detail in section 5). The irradiance cache is used during the rendering pass to determine the irradiance or incoming light at a geometric point in the scene. While photon mapping is capable of solving both direct and indirect illumination the preferred method of usage is to select a bidirectional reflectance distribution function (BRDF) algorithm for direct illumination such as Phong and utilize photon mapping for only the indirect illumination. Using photon mapping for both direct and indirect illumination would require significantly more photons in the scene to minimize the "blobs" that appear when few photons are available for computing an irradiance estimate (see figure 1).



When photon mapping is used for both direct and indirect illumination, the results can be very noisy. Even with 256k photons, this scene is still unable to provide smooth results given the reasonably sized search radius (defined later). This is why Phong shading makes an excellent choice for direct illumination, and photon mapping is reserved to compute indirect illumination only.

Figure 1. Photon mapping for direct and indirect illumination.

Using a direct illumination algorithm rather than photon mapping serves two purposes—it not only speeds up the rendering process, but it also reduces the amount of noise in the image. The full rendering equation is the sum of four components, which solve the outgoing radiance for a given geometric location in the scene. These components are derived from the three components that make up incoming radiance and the two terms that make up the BRDF. Together these equations form the basis for the full-rendering equation as described by Jensen (*3*).

$$
\begin{aligned}
L_r(x,\overrightarrow{\omega}) &= \int_\Omega f_r(x,\overrightarrow{\omega}',\overrightarrow{\omega})\,L_i(x,\overrightarrow{\omega}')\,(\overrightarrow{\omega}'\cdot\overrightarrow{n})\,d\overrightarrow{\omega}' \\
1 \qquad &= \int_\Omega f_r(x,\overrightarrow{\omega}',\overrightarrow{\omega})\,L_{i,l}(x,\overrightarrow{\omega}')\,(\overrightarrow{\omega}'\cdot\overrightarrow{n})\,d\overrightarrow{\omega}' \\
2 \qquad &+ \int_\Omega f_{r,S}(x,\overrightarrow{\omega}',\overrightarrow{\omega})\,(L_{i,c}(x,\overrightarrow{\omega}')+L_{i,d}(x,\overrightarrow{\omega}'))\,(\overrightarrow{\omega}'\cdot\overrightarrow{n})\,d\overrightarrow{\omega}' \\
3 \qquad &+ \int_\Omega f_{r,D}(x,\overrightarrow{\omega}',\overrightarrow{\omega})\,L_{i,c}(x,\overrightarrow{\omega}')\,(\overrightarrow{\omega}'\cdot\overrightarrow{n})\,d\overrightarrow{\omega}' \\
4 \qquad &+ \int_\Omega f_{r,D}(x,\overrightarrow{\omega}',\overrightarrow{\omega})\,L_{i,d}(x,\overrightarrow{\omega}')\,(\overrightarrow{\omega}'\cdot\overrightarrow{n})\,d\overrightarrow{\omega}' \;. \qquad (1)
\end{aligned}
$$

The full-rendering equation is an integral over the surface that determines the incoming radiance or final pixel value at that position in the scene. Each of the four integral components are computed numerically using a random sample set. The first component is the direct illumination from the BRDF, such as Phong. The second component accounts for the indirect illumination from caustics and specular reflections that ultimately gets stored in the caustics map. The third term is for caustic materials caused from reflective and refractive materials, which comprise the caustic photon map. The fourth term is the diffuse indirect illumination that comes from the irradiance cache.

3. The Photon Map

Photon mapping is filled with a plethora of parameters to control both its creation and how it is utilized in rendering. Automating these parameters by means of heuristics or other logical methods will prevent the end user from having to enter in values for each of these parameters. **[1]** The first parameter is the number of photons used in a scene, which is largely a balance between bias, memory, and speed. The number chosen should in some way reflect the complexity of the scene to maintain an acceptable level of detail in the photon map. For the case of rendering only triangles, one may choose to use an initial value of 10,000 photons and an additional 1 photon for every 100 triangles in the scene.

While generating the photon map, photons will be emitted from a designated set of light sources and stored in a data structure such as a one-dimensional array. Ultimately, the photons will be organized into a kd-tree* data structure that gets used to generate a new kd-tree known as the irradiance cache. The result of this pass will be a single kd-tree filled with irradiance cache points, a separate caustic photon map, and, optionally, a separate shadow photon map. The initial structure containing the diffuse photons will be discarded after the irradiance cache has been built. **[2]** The kd-tree is preferred over other data structures such as an octree, bsp, or

* Defined in section 4.

voronoi graph because Jensen (*3*) has shown that the kd-tree exhibits the most suitable properties for photon mapping.

3.1 Emitting Photons

Photons are emitted from a set of designated light sources. Unlike real photons that have a static energy, the energy of these virtual photons will be dynamic. Simulating the quantum physics of light with real photons would require a tremendous amount of memory and computational power, thus making it impractical for today's desktop computer hardware. Therefore, a discrete number of photons are used, typically many orders of magnitude less than a simulation with real photons. **[3]** As photons are generated for each light source, be sure to treat the light sources as invisible geometry so that photons do not get trapped inside.

The energy of these virtual photons are typically represented by an RGB triplet or wavelength value. The number of photons used in the scene is typically supplied as a user-definable value. The photons start with the initial energy of the light source divided by the total number of photons being emitted from the light source. Photons can either be emitted from a set of random points on the light source or from a single point representing the center of the light source. The ray for each photon can be generated using a uniform rejection sampling technique where three floating values representing the (x, y, z) components of the direction vector are randomly generated inside of a conditional loop that does not exit until the sum of all three values is less than or equal to one. Once all of the photons have been emitted from all of the light sources, one may wish to use a scaling factor to adjust the energy of each photon to increase or decrease the global illumination contribution.

The data structure of the photon is very important when dealing with large numbers of photons. Typical scenes may involve many thousands to millions of photons. If a scene utilizes 1 million photons and the data structure for each photon is 32 bytes, then at least 32-million bytes of memory will be required for the photon map. As the number of photons approaches infinity, the bias approaches 0. The photon structure will typically consist of at least a position, normal, and power. The position will typically be 3 floating values or 12 bytes. The normal can consists of a short, thus giving it 65,536 unique values, which is 2 bytes. The power can consist of 3 floats, which is another 12 bytes. Altogether, the base structure size is 26 bytes. **[4]** Generating a photon structure that is 32, 48, or 64 bytes should be considered since most processor architectures operate on 32 or 64 byte cache lines. If the photon structure is not one of these sizes, then try padding it; this will reduce the amount of cache lines required to get the data into processor cache and therefore give a noticeable performance boost.

3.2 Photon Propogation

Photons are emitted from a set of light sources and will propagate through the scene as a function of the material with which they are interacting. This process will generate the photons used in the diffuse, caustic, and shadow photon maps. With each diffuse interaction, the photons will

have a probability of either continuing to propagate or being absorbed. Emitting photons in this manner will permit a uniform distribution of photons being deposited throughout the geometry. Additionally, one will note that the photon energies are diminishing with each propagation, with the exception of materials having values of 1.0 for a normalized RGB value. As the number of propagations a photon undergoes increases, the lower the probability it has of continuing. This implicit property dictates that photons having near zero energies will have a lower probability of existing in the scene than photons with higher energy values.

As photons are emitted, they should be stored in a linear array or similar data structure once they finish propagating in the scene. Once all of the photons are done emitting, they should get stored in a kd-tree, using the linear array as input, i.e., the photon map. The kd-tree is best suited for storing the photons because of its efficient partitioning and lookup characteristics as shown by Jensen (*3*).

For reflective and refractive materials, i.e., caustics, photons will either reflect off of or transmit through the surface at each intersection point with respect to the parameters assigned to the material with which the photon is interacting. When the material is partially diffuse, the method of terminating and spawning new photons still exists, except that the probability of terminating is multiplied by the diffuse fraction of the material. As is the case with path tracing, in photon mapping, while each photon propagates through the scene, the energy is multiplied with the material energy at each intersection. For example, let a photon propagate three times in a diffuse scene before terminating, where all geometry has an RGB value of (0.8, 0.8, 0.8). For simplicity, let the initial value of the photon be (1.0, 1.0, 1.0). As the photon propagates, the values of the photon will be (0.8, 0.8, 0.8), (0.64, 0.64, 0.64), and (0.512, 0.512, 0.512), respectively.

Photon mapping uses a method of Russian roulette to reduce computational costs and storage requirements. The probability that a photon will continue to propagate decreases as the propagation depth increases. For materials that have a diffusely reflective value R [0–1.0], one can choose to reflect N photons with R power, or reflect N/R photons and have 1-N/R photons absorbed during propagation. **[5]** Clearly, this second method requires less reflection computations and photon propagations. For the case of both specular and diffuse reflections, one can use the following function (*3*) to control the reflection and absorption probabilities for various materials where the value of ξ is a randomly generated number between 0 and 1.

$$\xi \in [0, P_d] \rightarrow \text{diffuse reflection.} \tag{2}$$

$$\xi \in [P_d, P_s + P_d] \rightarrow \text{specular reflection.} \tag{3}$$

$$\xi \in [P_s + P_d, 1] \rightarrow \text{absorption.} \tag{4}$$

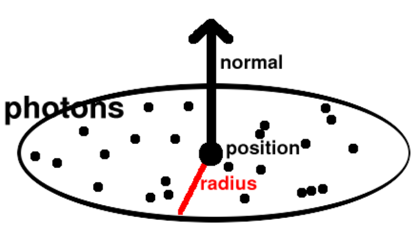
A new value for ξ is calculated every time the photon propagates. As the number of propagations increases, the probability that the photon will be absorbed increases. Once the photon is absorbed, it is stored in the appropriate photon map data structure.

3.3 Separate Photon Maps—Better Caustics and Shadows

The use of separate photon maps is an additional optimization that should be implemented for improved speed because less photons will be required to achieve the same results as using a single photon map. Caustic photons from reflections and refractions will typically be the result of some focusing or reflecting medium, which results in a sharp lighting feature, such as the focusing of light through a glass sphere or the cardioid formed on the inner circle of a reflective ring. Because photons in caustics generally have sharp features, the use of a small search radius in the caustic lookup is paramount. Diffuse photon map lookups generally require a larger search radius so that one can obtain smooth results. Both diffuse and caustic photons do not belong in the same photon map. The caustic photon map should only contain caustic photons, and the diffuse photon map should contain diffusely propagated photons.

Shadow photons are used for creating a smooth shadow region. A shadow photon has a negative energy, which is used in an additive process to subtract energy from the irradiance estimate. Shadow photons can exist in their own kd-tree and are usually created during the photon-emitting phase. When performing an irradiance estimate during the rendering pass, one can examine the shadow photon map to find the shadow photons in the local area. Shadow photons are not an essential part of photon mapping, but they are a definite optimization to an already functional implementation since a shadow photon map lookup is less costly than firing a large number of shadow rays.

When performing a kd-tree lookup, one must define a search radius that will be used to determine the results which will be accepted or rejected. This search radius is either a circular or elliptical region that extends outward from the point where an estimate is being calculated. The developer may choose to make the size of the search radius (see figure 2) a heuristic that is a function of the scene's size or a control that is made available on the graphical user interface.



Each kd-tree lookup requires a position, normal, and search radius to gather a set of N photons. During the rendering pass, the kd-tree lookup for the irradiance cache is identical to the kd-tree lookup for the photon map except that the irradiance cache kd-tree does not contain photons; it contains irradiance cache points.

Figure 2. A kd-tree lookup.

4. The kd-tree

The kd-tree is an optimal data structure for storing photons and irradiance cache points in a scene for quick retrieval. Alternative data structures such as an octree or bsp will work, but the author found they did not perform as well as the kd-tree. Understanding the mechanics of the kd-tree is rather important for selecting an optimal median or pivoting algorithm, which is used to determine the position to split each cell, thus making the kd-tree as balanced as possible. Each parent node in the kd-tree consists of two child nodes, an axis-aligned splitting plane, and the axis coordinate where the plane is being split. Once the photon map and the irradiance cache have been constructed, the original list of diffuse photons can be discarded. With a kd-tree, one can efficiently locate any elements in O(log n) time using a supplied position, normal, and search radius. Since the position of each photon is three-dimensional in nature, so too is the kd-tree used in photon mapping. For a detailed description of kd-trees, see Andrew Moore's "An Introductory Tutorial on kd-trees" (*4*). For information on implementing photon mapping on a GPU, see Purcell et al. (*5*), where the use of an alternative data structure used with the GPU as well as some of the difficulties encountered in implementing the photon mapping algorithm in this more restrictive programming environment is described.

5. Irradiance Cache

While the photon map is suitable for determining the global illumination in a scene, there is more information that can be extracted from the photon map to further reduce computational costs. With only the photon map, one would be required to sample the irradiance using many photon map lookups during the render pass. Each lookup involves firing a bundle of rays from the eye rays intersection point and then finding a set of nearest neighbor photons at the intersection point using the photon map kd-tree. This process would be $O(n^2 \log n)$ since M sample rays would be fired to do N photon lookups into an O(log n) kd-tree lookup. If one precomputes the irradiance for a set of points in the scene, the algorithm becomes a significantly faster O(log n) kd-tree lookup.

There are two approaches one can take when computing the irradiance cache. The first approach involves computing the irradiance cache points as they are needed during the render pass. Jensen (*3*) suggests the following formula for computing irradiance, which uses previously computed irradiance values to interpolate a new irradiance value at the position x given normal n.

$$E(x, \vec{n}) \approx \frac{\sum_{i,\omega_i > 1/a} \omega_i (x, \vec{n}) E_i (x_i)}{\sum_{i,\omega_i > 1/a} \omega_i (x, \vec{n})}. \tag{5}$$

If there are no previously computed irradiance values that satisfy the condition $\omega_i > 1/a$, then a new value is computed. For computing the weight ω_i Jensen (*3*) uses the following equation where R_0 is the harmonic mean distance from x_i.

$$\omega_i = \frac{1}{\frac{\| x_i - x \|}{R_o} + \sqrt{1 - 1 \vec{n} \cdot \vec{n}_i}}. \tag{6}$$

The fraction 1/a is a user-definable parameter to control how much deviation can occur using the previously computed irradiance values for interpolation until the amount of error surpasses the allowable threshold.

[6] The second approach involves less work by precomputing the irradiance, suggested by Larsen and Christensen (*6*). The selection of irradiance points is not as optimal as the first approach, but the algorithm is faster and less sophisticated. The concept involves selecting a subset of the photon positions such as every fourth photon and computing the irradiance at that point. These points, in turn, become the irradiance cache points. During the render pass, the irradiance estimate involves locating a nearest irradiance cache point or group of points if preferred.

The stochastic uniform sampling of a tesselated hemisphere is used to generate the sample rays which, in turn, are used to calculate the irradiance at each of the suitable irradiance cache points. Each of the sample rays are fired into the scene to obtain an intersection point where a nearest neighbor lookup into the photon map is performed to obtain a mean radiance. The radiance estimates from each ray are used to calculate an irradiance estimate. Figure 3 illustrates an example of the irradiance cache points generated using the irradiance method described by Jensen (*3*). Figure 4 illustrates a scene rendered using the irradiance cache in figure 3.

For each irradiance cache point, a virtual unit hemisphere aligned to the surface normal is used as a template for generating a set of cells. The unit hemisphere is tesselated into M × N cells, and for each cell, a ray is fired through the center of it plus a random delta. The delta is simply a random value between 0 and half the size of the cell that is added to each ray direction vector. For each generated ray, a diffuse photon map lookup occurs so that a radiance estimate can be calculated. Once all of the radiance estimates are finished, they are averaged to form the final irradiance estimate for each cache point, which is used during the rendering pass. Figure 5 illustrates an example of the ray generation process using a virtual tesselated unit hemisphere.

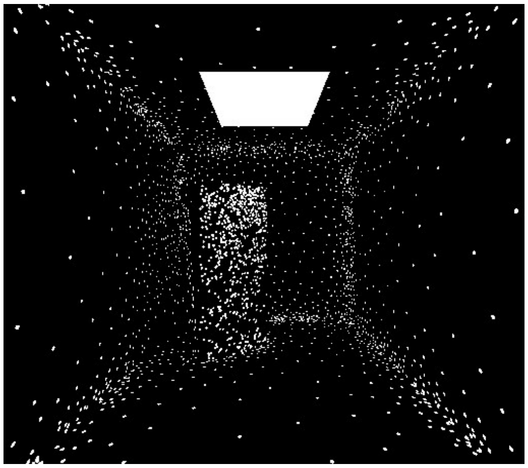


Figure 3. Irradiance cache is represented by white dots. Notice the point density is low in areas where normals are not changing.

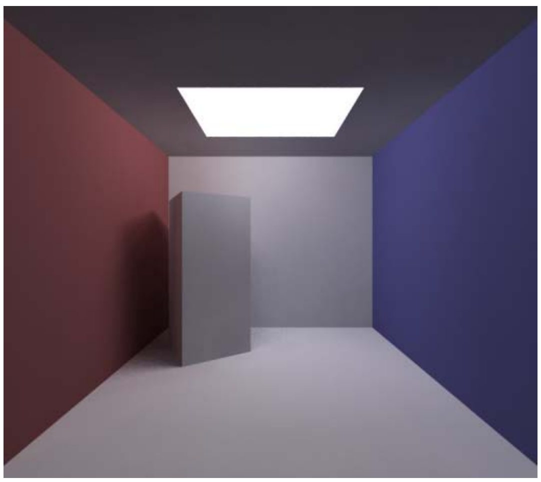
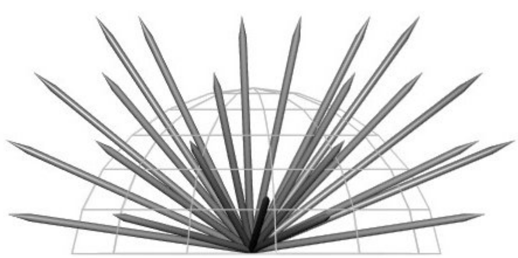


Figure 4. Scene rendered with photon mapping using the irradiance cache from figure 3.

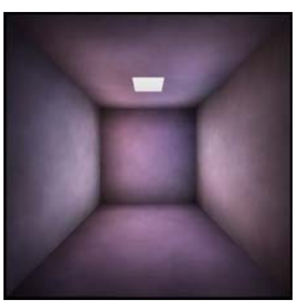


A tesselated unit hemisphere where M = 13 and N = 6. A single ray is fired stochastically through each cell to return a radiance estimate. **[7]** Radiance estimates from each ray are averaged together to form an irradiance estimate. Each ray has a random perturbation to the direction vector, allowing it to shoot anywhere through its designated cell region.

Figure 5. Virtual tesselated unit hemisphere.

Several heuristics must be employed in order to obtain satisfactory radiance estimates for each irradiance calculation. While the kd-tree for the diffuse photon map is traversed, one must compare the normal of each candidate photon to the surface normal at the intersection point to determine if the photon is within an angular tolerance to be accepted or rejected in the estimate. For the case of a sphere, the intersection normal will almost always be different than neighboring photons on the surface. Therefore, one must choose an appropriate tolerance angle such that those neighboring photons are accepted; 60° typically works well. For the case where two perpendicular planes meet, such as a wall connected to the floor, one must not accept photons from the wall if the estimate is being done for the floor; thus, the angular tolerance must be less than 90°. Divide the sum of the located photon energies by the area of the search radius to find a mean estimated energy for the given surface position.

When performing a lookup into the photon map, one must supply a search radius to determine which photons from the resulting search will be accepted or rejected. For the case where a search is being performed near the boundary where two edges lie perpendicular to one another, the search radius will extend beyond the edge of the given surface. **[8]** In the event that this problem occurs, one must tally the accepted and rejected photons from the search in order to create a correction factor. The energy of the accepted photons will either be averaged or processed through a filter so that one can obtain an energy estimate. On edge boundaries, some photons will be rejected during the search process, resulting in a lower-than-expected energy estimate (see figure 6). One must divide the resulting energy estimate by the ratio of accepted photons to the sum of accepted and rejected photons. Doing so will interpolate what the energy in this area should have actually been, thus removing any dark edges or corners that may appear. This phenomenon also occurs during the lookup into the irradiance cache as the mechanics are identical to performing a lookup into the photon map.



Dark corners are often the result of a search radius that extends beyond the edge of the geometry. When this occurs, the estimate picks up fewer photons than it expects. Thus, when the results are summed and divided by the area, the estimate ends up smaller than it should be. A correction factor can be calculated to remedy this problem. No correction factor was applied to this scene, hence the dark edges. The correction factor is calculated and applied to both photon map lookups and irradiance cache lookups. One should use the same code for both photon map lookups and irradiance cache lookups since the algorithm for both processes are nearly identical.

Figure 6. Scene with dark edges, resulting from no correction factor.

The irradiance cache algorithm requires very little work to parallelize. For multiprocessor systems, one can expect to see a linear speedup in the time it takes to build the irradiance cache. All that is required to parallelize the irradiance caching algorithm is a single semaphore. The semaphore is used to protect the irradiance cache list from being written to by more than one

thread at a time. The photon-emitting algorithm can also be parallelized, but the speedup is negligible since the vast majority of preparation time is spent building the irradiance cache.

6. Optimizations

There are various optimizations one can employ to speed up the entire photon-mapping process. First, one should precompute the irradiance cache so that this process becomes a prerendering pass; this ensures unnecessary processing is not done during the rendering pass. Second, the use of separate photon maps should be employed so that one can still achieve sharp caustics, soft shadows, and a minimal number of photons for indirect illumination. Third, importance mapping should be implemented to reduce the number of photons required in a scene, thus saving on memory. Fourth, one should implement the use of multiple photon maps for irradiance estimates to improve performance.

6.1 Importance Mapping

Importance mapping is a method of shooting *importons* from the camera into the scene to reduce the amount of photons and irradiance cache points needed in a scene. This process changes photon mapping from a view-independent to a view-dependent algorithm. For scenes with many lights and nonvisible partitions, there can be a tremendous benefit. For example, envision an office building that contains hundreds of rooms, each containing a ceiling light, and most of which are not visible from the camera. Photon mapping will naïvely shoot photons from every light source and deposit photons throughout the entire building. One does not need photons stored in any of the rooms that have no visual impact on the rooms currently in view. Importons are deposited in the same manner that photons are. Finally, one must march through each of the importons and mark any photons within its location as important. After marching through all of the importons and marking important photons, any remaining photons in the diffuse photon map should be discarded.

6.2 Multiple Photon Maps for Irradiance Estimates

It has been shown that various rejection methods must be employed to prevent an estimate from being contaminated by nearby photons. As a result, Larsen and Christensen (*6*) have shown that using multiple photon maps can be employed for the irradiance estimate to improve performance during the prerendering pass. The idea is to use different photon maps on adjacent surfaces where there is a sufficiently large angle between them as a way of preventing contamination. The result of this approach is a collection of separate photon maps assigned to each different surface. During the building of the irradiance cache, one would then need to perform a lookup into multiple photon maps instead of just one photon map containing all of the photons for the diffuse indirect illumination.

7. Rendering

Rendering with photon mapping is a rather straightforward process that involves computing each of the four components in the full rendering equation. Direct lighting is computed using a BRDF such as an attenuated version of Phong shading. The irradiance cache is used to sample the irradiance at the geometric location being rendered. The caustics photon map, which contains transmissive and reflective photons, and any other additional photon maps, i.e., the shadow photon map, are used to compute global illumination. The sum of these four components will be the value of the resulting pixel.

7.1 Direct Illumination

Two popular algorithms for computing direct illumination are Phong (*7*) and Goraud shading (*2*). **[9]** It is imperative that the direct-illumination algorithm have an attenuation coefficient so that energy decreases as a function of the distance in the same manner that occurs in photon mapping. The denominator of the attenuation coefficient in Phong is typically of the form a + bd + cd^2 to prevent dividing by zero and to provide some linear scaling control. One can alternatively choose to make the denominator p + d^2 where p is defined as a very small number and d is the distance. Östberg and Thorin (*8*) illustrate an attenuated version of Phong.

$$L = [\frac{1}{a + bd + cd^2}](k_d L_d 1 \cdot N + k_s L_s (r \cdot v)^{\alpha}) + k_a L_a \,. \tag{7}$$

[10] In addition to including attenuation in the direct-illumination model, it is as equally important not to include the ambient term. Ambient contributions are not used in any of the direct illumination models because the intent of ambient lighting is to cheaply simulate indirect illumination. Since photon mapping is used for indirect illumination, the ambient term should not be used.

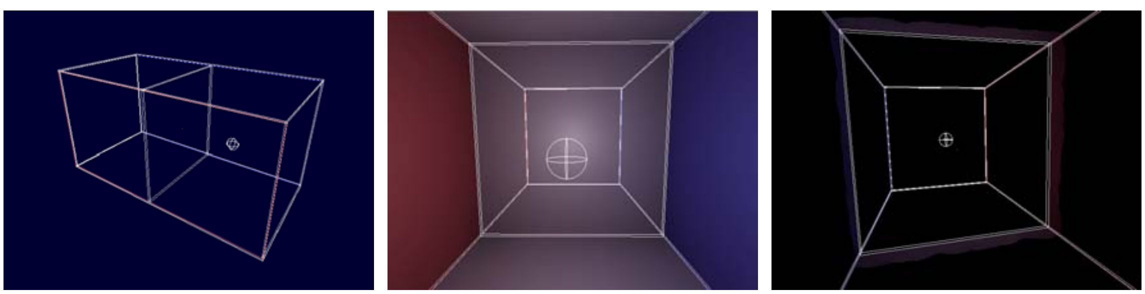
7.2 Global Illumination

The second half of the rendering pass involves finding the indirect illumination from the irradiance cache. For each ray fired from the camera, an irradiance cache lookup will take place to find a set of nearby irradiance cache points. The number for the set can be user definable or a constant such as 50. When the irradiance lookup algorithm returns with a set of points, the algorithm can then either average the results or process them through a filter. The purpose of the filter is to further improve the quality of the sample set found. Some of the more common filters are the gauss and cone filters, which give the irradiance cache points closest to the intersection point the most influence. In addition to calculating the indirect illumination from the irradiance cache, the contributions from the caustic and shadow photon maps should also be added to the result. The net result of this part of the rendering pass is a pixel value that includes the indirect

diffuse, caustic, and shadow contributions. This result is added to the direct-illumination calculation to generate the final pixel value.

8. Degeneracies

There are several degeneracies in photon mapping that require some additional nontrivial code to fix. **[11]** The first degeneracy exists when one geometric plane bisects another, forming the “T” degeneracy. In the irradiance estimate during the rendering pass, one provides the irradiance lookup function with the number of photons to use as an estimate, a search radius, a normal, and a geometric location. If photons have been stored on the side of a bisected plane that receives no direct light, the irradiance lookup algorithm will locate photons from the side receiving direct light when doing an irradiance estimate near the corner of the side that is not receiving direct light. (There is simply not enough information available to the irradiance lookup algorithm to prevent this degeneracy from occurring without introducing more overhead into the irradiance estimate code to make some intelligent decisions about what irradiance cache points to accept or reject based on various geometric partitions that exist in a scene.) Figures 7 and 8 illustrate this “T” degeneracy. This problem could be corrected by partitioning the geometry where planes meet at a user-defined angle or by creating an algorithm to fire several rays to determine whether a partition is nearby; both are nontrivial to implement.



(a) A wire-frame view of the entire scene. One room contains a light source, and the other is completely isolated from light.

(b) A rendering of the room containing the light source.

(c) A rendering from the unlit room. Notice that the algorithm is gathering photons around the edges of the back wall.

Figure 7. “T” degeneracy room views.

[12] A second degeneracy exists when the scene is being illuminated by a small crevice from another room with sufficient lighting. A scene where this problem exists is one which contains two rooms that are separated by a door with a keyhole, and the only light the unlit room receives is from the keyhole and through the crack on the bottom of the door. Because there is only a small area through which photons may propagate, one must use a large number of photons to render this scene accurately. Metropolis Light Transport handles this problem with its use of

(a)

A path-traced reference image exhibiting correct illumination above the counter. Rendered with 8192 rays per pixel using ADRT.

(b)

A photon-mapped rendering of the same scene. Notice that the area above the counter is darker than it should be. Rendered with 16,384 photons using BRL-CAD.

Figure 8. "T" degeneracy images.

various mutation strategies (*9*), which sample the scene based on the probability a ray will reach a light source. Path tracing uses a large number of samples; therefore, this does not become an issue. Photon mapping requires an importance map.

[13] A third type of degeneracy occurs in the case of simulating motion blur (*10*). A single photon map cannot be reused for each frame of an animation that contains dynamic geometry. The use of a temporal photon map must be used to accommodate the dynamic global illumination. This approach requires a four-dimensional kd-tree in which photons have a time parameter so that when a lookup occurs, only the photons from that temporal region are included in the estimate. With this method, one can interpolate between temporal regions without requiring a new photon map for each frame.

9. The Big Picture

Now that all of the finer details of photon mapping have been presented, it is important to understand how all of the various parts of photon mapping fit together to form the finished product. Again, think of photon mapping as a two-pass algorithm. During the first pass, a bunch of photons are emitted from all of the light sources into the scene. Importance mapping is used to optimize this part of the first pass to reduce the amount of photons needed. The photons propagate through the scene and are stored in their respective diffuse, caustic, or shadow photon lists. Each list of photons is used to construct a kd-tree, i.e., the photon maps. Next, the diffuse photon map is used to generate the irradiance cache by selecting a subset of the photons in the scene as irradiance cache points, whereby a virtual tesselated hemisphere is used to shoot out a

number of sample rays to compute the irradiance. Once the first pass is complete, there should exist an irradiance cache, a caustic photon map, and a shadow photon map if implemented.

During the second pass, the rendering pass, rays are shot from the camera into the scene. Each time a ray from the camera intersects a piece of geometry in the scene, the full lighting equation is evaluated. The direct lighting is computed using Phong, and an irradiance lookup is done to determine indirect illumination at that point. Finally, the caustic and shadow photon maps perform a lookup. The results are summed together to form the final pixel value, and the step is repeated until the image is completely rendered.

10. Summary

Different applications have different requirements for global illumination. Photon mapping is an excellent addition to any rendering system where the overall objective is to increase realism by the addition of global illumination. The photon-mapping algorithm may be a nontrivial effort to implement. One will have to find a balance between the controls in photon mapping that will be managed by heuristics and those that will be exposed in the user interface. Providing the end user with too many controls results in a nonoptimal interface. Dealing with the various degeneracies is not an easy feat either, but each of those can be dealt with as the problem arises. Photon mapping produces some very realistic looking images when implemented and used correctly (see figure 9).

However, if the underlying objective for the developer is to produce unbiased realistic images by means of a turn-key solution, then they may wish to exercise other global illumination algorithms such as path tracing. While path tracing requires approximately two orders of magnitude more processing time than photon mapping, there are certain techniques to bring the path-traced rendering process up to the speed of the photon-mapped rendering process. If one has not already done so, one may wish to improve the performance of one's ray-tracing engine by an order of magnitude via the work done by Wald et al. (*11*). This improvement will, in turn, make path tracing only an order of magnitude slower than photon mapping. With a small cluster of networked computers and distributed capabilities in the ray tracer, one can expect path tracing to now perform on par with photon mapping, but in a turn-key environment. Path tracing typically requires 50 lines of code, which makes the implementation process quick and the code base easy to maintain. With this in mind, one may choose to implement a high-performance distributed ray-tracing engine with path tracing.

The top left image represents standard Phong rendering without the ambient term. The top right image shows Phong with a constant 0.4 value for ambient light. The bottom left image uses phong for direct illumination and photon mapping for indirect illumination; notice the subtle color bleeding on the background and on the tree limbs. The bottom right image is the net result of what photon mapping generated for this image. This result is simply added to the top left image to generate the bottom left image.

Figure 9. Tree scene before and after photon mapping.

11. References

1. Jensen, H. W.; Christensen, N. J. Photon Maps in Bidirectional Monte Carlo Ray Tracing of Complex Objects. *Computers & Graphics* **1995**, *19* (2), 215–224.
2. Watt, A.; Watt, M. *Advanced Animation and Rendering Techniques*; Pearson Education Limited: Essex, England, 1992; pp 23, 293–295.
3. Jensen, H. W. *Realistic Image Synthesis Using Photon Mapping;* A K Peters Ltd.: Natick, MA, 2001; p 97.
4. Moore, A. Efficient Memory-based Learning for Robot Control. Ph.D. Thesis, University of Cambridge, 1991.
5. Purcell, T. J.; Donner, C.; Cammarano, M.; Jensen, H. W.; Hanrahan, P. Photon Mapping on Programmable Graphics Hardware. Stanford University, Stanford, CA; *Graphics Hardware* **2003**.
6. Larsen, B. D.; Christensen, N. J. Optimizing Photon Mapping Using Multiple Photon Maps for Irradiance Estimates. *WSCG Proceedings*, Plzen, Czech Republic, 3–7 February 2003.
7. Glassner, A. S. *Principles of Digital Image Synthesis*; Morgan Kaufmann Publishers, Inc.: San Francisco, CA, 1995; Vol. 2, pp 726–728.
8. Östberg, B.; Thorin, H. Lights. http://www2.hh.se/staff/jovall/dg04/elevwebb/light /Seminar/Lights_and_the_Phong_model.htm (accessed March 2005).
9. Veach, E.; Guibas, L. J. Metropolis Light Transport. *Siggraph 97 Proceedings*; Addison-Wesley: Reading, MA, August 1997.
10. Cammarano, M.; Jensen, H. W. Time Dependent Photon Mapping. *Thirteenth Eurographics Workshop on Rendering*, Department of Computer Science, Stanford University, Stanford, CA, 2002.
11. Wald, I.; Slusallek, P.; Benthin, C.; Wagner, M. Interactive Rendering with Coherent Ray Tracing. Computer Graphics Group, Saarland University; *Eurographics* **2001**.

NO. OF COPIES	ORGANIZATION
1 (PDF ONLY)	DEFENSE TECHNICAL INFORMATION CTR DTIC OCA 8725 JOHN J KINGMAN RD STE 0944 FORT BELVOIR VA 22060-6218
1	US ARMY RSRCH DEV & ENGRG CMD SYSTEMS OF SYSTEMS INTEGRATION AMSRD SS T 6000 6TH ST STE 100 FORT BELVOIR VA 22060-5608
1	INST FOR ADVNCD TCHNLGY THE UNIV OF TEXAS AT AUSTIN 3925 W BRAKER LN STE 400 AUSTIN TX 78759-5316
1	DIRECTOR US ARMY RESEARCH LAB IMNE ALC IMS 2800 POWDER MILL RD ADELPHI MD 20783-1197
3	DIRECTOR US ARMY RESEARCH LAB AMSRD ARL CI OK TL 2800 POWDER MILL RD ADELPHI MD 20783-1197
3	DIRECTOR US ARMY RESEARCH LAB AMSRD ARL CS IS T 2800 POWDER MILL RD ADELPHI MD 20783-1197
	ABERDEEN PROVING GROUND
1	DIR USARL AMSRD ARL CI OK TP (BLDG 4600)

NO. OF COPIES	ORGANIZATION
1	DIRECTOR FORCE DEV DAPR FDZ RM 3A522 460 ARMY PENTAGON WASHINGTON DC 20310-0460
1	US ARMY TRADOC ANL CTR ATRC W A KEINTZ WSMR NM 88002-5502
1	USARL AMSRD ARL SL EA R FLORES WSMR NM 88002-5513
1	USARL AMSRD ARL SL EI J NOWAK FORT MONMOUTH NJ 07703-5601
	ABERDEEN PROVING GROUND
1	US ARMY DEV TEST COM CSTE DTC TT T APG MD 21005-5055
1	US ARMY EVALUATION CTR CSTE AEC SVE R BOWEN 4120 SUSQUEHANNA AVE APG MD 21005-3013
1	US ARMY EVALUATION CTR CSTE AEC SVE S R POLIMADEI 4120 SUSQUEHANNA AVE APG MD 21005-3013
1	US ARMY EVALUATION CTR CSTE AEC SV L R LAUGHMAN 4120 SUSQUEHANNA AVE APG MD 21005-3013
12	DIR USARL AMSRD ARL SL J BEILFUSS P TANENBAUM AMSRD ARL SL B J FRANZ M PERRY AMSRD ARL SL BB D BELY D FARENWALD S JUARASCIO AMSRD ARL SL BD R GROTE AMSRD ARL SL BE L ROACH AMSRD ARL SL E M STARKS AMSRD ARL SL EC J FEENEY E PANUSKA

INTENTIONALLY LEFT BLANK.